Development Framework for Web Service Choreographies in Pervasive Environments

Gregory Van Seghbroeck

Supervisors: Bart Dhoedt, Filip De Turck

I. Introduction

Nowadays mobile devices like PDAs and smart phones are not only becoming extremely popular, mainly due to their decreased price tag, but they are also becoming more intelligent. These mobile devices are a valuable asset to incorporate in large scale applications, e.g. making bank payments via SMS. However, it does not have to stop with mobile devices and server applications. When we add sensor networks, network-aware home appliances, etc to the mix, you cannot imagine anything that cannot be created for these pervasive, heterogeneous environments. The sky is the limit!

One of the many types of service oriented architectures (SOA) that can be used to create these pervasive applications are service choreographies. Designing choreographies, not to mention executing one, is not a sinecure. That is why we present the necessary building blocks for a development framework to facilitate creating such choreographies and in particular web service choreographies. The development framework includes all the different stages of the development cycle: from design over validation to deployment. It will mask the complexities of the validation and the projection steps, taking into account that the service choreography will include besides servers, sensors and similar limited devices. This can potentially pose problems during deployment, since these devices very often have few computation power and small embedded memory, so the used algorithms have to be fast with a small memory footprint.

Figure 1 depicts the flow and the different building blocks of the development framework. It is roughly divided into two phases, an implementation and a deployment phase. Throughout the flow different models are used tot represent the data and where it is possible we use real standards. Most of the steps of the framework can be automated; others still require some user interaction.

II. The Implementation Phase

The implementation phase first of all starts with the design of the global choreography. We use W3C's WS-CDL specification to describe the choreography. In [1], the authors

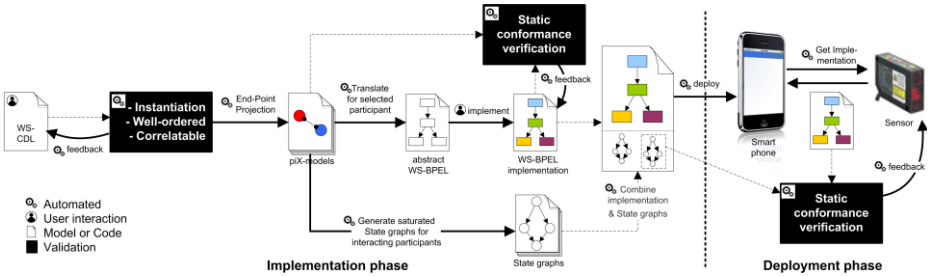

Figure 1 Development framework flow

proved the relation between WS-CDL and the π-calculus, a well known process algebra used to model mobile distributed systems. The fact that WS-CDL has important similarities with the π-calculus, helps us with designing interesting validation mechanisms. E.g. the channel instantiation validation, described in [2], makes use of WS-CDL's notion of channels. The other validations that need to be performed on the choreography description are correlatability (described in [3]) and well-orderedness. All these validations make sure that the described choreography can unambiguously be executed. Via the End-point Projection technique presented in [1] we translate the channel instances, created during the instantiation validation, to an intermediate model, the piX-model. This model, first presented in [4], models the behavioural aspects of the channel instances. In the following step the different piX-models are translated to abstract WS-BPEL stubs. These rather difficult steps of the implementation phase can be fully automated.

We now have a bunch of abstract WS-BPEL stubs that represent the behavioural aspects of the choreography, but these stubs need to be implemented further to meet the specific needs of the domain and the device. This part of the implementation phase is entirely up to the developer of that specific device. We use WS-BPEL in our development framework, but this can be any programming language capable of running small workflows (even JAVA or C). The only thing we need to do additionally is define a mapping between the piX-model and the chosen language and vice versa.

When the implementation is finished, the Static Conformance Verification (SCV) method can be used to verify whether the implementation still is conformant to the choreography description. This validation is thoroughly described in [4]. It uses the piX-model and Saturated State Graphs (SSG) as input. An SSG is a labelled graph using the behavioural activities as labels and can be derived from the piX-model. While deploying the implementation to its respective device, we will send, instead of the entire WS-CDL description, all the other choreography partners' SSGs to the device as well. There is no problem in doing so, because all these state graphs combined, exactly represent the behaviour of the choreography.

III. The Deployment Phase

When we deploy the implementation to its device, we will also verify the conformance of the other choreography partners to be sure that they behave as described in the WS-CDL. Each partner's implementation will be retrieved over the network. These implementations are then translated to the piX-model and used together with the deployed state graphs as input for the SCV. Since the algorithm now is running on the device itself, we will benefit from the efforts taken to reduce its complexity:

- piX-models will be as small as possible, due to the channel instantiation [2];
- SSGs are already created during the implementation phase and deployed together with the implementation;
- By using the piX-model the SCV is drastically reduced in complexity [4].

ACKNOWLEDGEMENT

Gregory Van Seghbroeck acknowledges the IWT for a PhD research grant.

References

[1] M. Carbone, et all, "A Theoretical Basis of Communication-Centered Concurrent Programming", 2006.

[2] G. Van Seghbroeck, et all, "Automated instantiation and extraction of web service choreographies", ICIW, May 2009

[3] G. Van Seghbroeck, et all, "Message Correlation in Web Services Choreographies: a 4-phase Validation Method", ECOWS, *accepted*, November 2009.

[4] G. Van Seghbroeck, et all, "Web service choreography conformance verification in M2M systems through the piX-model", SIPE, July 2007.